This study analyzes chronic kidney disease using machine learning techniques based on a chronic kidney disease (CKD).

Chronic Kidney Disease (CKD) or chronic renal disease has become a major issue with a steady growth rate. A person can only survive without kidneys for an average time of 18 days, which makes a huge demand for a kidney transplant and Dialysis. It is important to have effective methods for early prediction of CKD. Machine learning methods are effective in CKD prediction. This work proposes a workflow to predict CKD status based on clinical data, incorporating data prepossessing, a missing value handling method with collaborative filtering and attributes selection

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force
```

jupyter nbextension install --py luxwidget jupyter nbextension enable --py luxwidget pip install lux-api

```
# import necessary packages

import os
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
import sklearn
import warnings
warnings.filterwarnings('ignore')
#import lux # EDA
from sklearn.preprocessing import MinMaxScaler
```

```
# import the dataset
df = pd.read_csv('/content/drive/MyDrive/Colab_Notebooks/DL/Kidney Disease Health Care/kidney_disease.csv')
df.head(n=10)
```

	id	age	bp	sg	al	su	rbc	pc	pcc	b
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpreser
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpreser
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpreser

```
1 df.shape
```

```
(400, 26)
```

```
1 # Total count of null values
2 df.isnull().sum()
```

```
id                 0
age                9
bp                12
sg                47
al                46
su                49
rbc              152
pc                65
pcc                4
ba                 4
bgr               44
bu                19
sc                17
sod               87
pot               88
hemo              52
pcv               70
wc               105
rc               130
htn                2
dm                 2
cad                2
appet              1
pe                 1
ane                1
classification     0
dtype: int64
```

```
1 # Total Null count in percentage
2 df.isnull().sum() / len(df) *100
```

```
id                 0.00
age                2.25
bp                 3.00
sg                11.75
al                11.50
su                12.25
rbc               38.00
pc                16.25
pcc                1.00
ba                 1.00
bgr               11.00
bu                 4.75
sc                 4.25
sod               21.75
pot               22.00
hemo              13.00
pcv               17.50
wc                26.25
rc                32.50
htn                0.50
dm                 0.50
cad                0.50
appet              0.25
pe                 0.25
ane                0.25
```

```
classification     0.00
dtype: float64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    int64  
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object 
 7   pc              335 non-null    object 
 8   pcc             396 non-null    object 
 9   ba              396 non-null    object 
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object 
 17  wc              295 non-null    object 
 18  rc              270 non-null    object 
 19  htn             398 non-null    object 
 20  dm              398 non-null    object 
 21  cad             398 non-null    object 
 22  appet           399 non-null    object 
 23  pe              399 non-null    object 
 24  ane             399 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

Handling missing values using SimpleImputer

```
# sklearn approach to handle all missing data at one go

from sklearn.impute import SimpleImputer
imp_mode = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df_imputed = pd.DataFrame(imp_mode.fit_transform(df))
df_imputed.columns = df.columns
df_imputed
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appe
0	0	48.0	80.0	1.02	1.0	0.0	normal	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	goo
1	1	7.0	50.0	1.02	4.0	0.0	normal	normal	notpresent	notpresent	...	38	6000	5.2	no	no	no	goo
2	2	62.0	80.0	1.01	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	5.2	no	yes	no	po
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Imputed columns

```python
df_imputed.columns
```

```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
       'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

```python
# After handling missing data, there is no null values
df_imputed.isnull().sum()
```

```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               0
dm                0
cad               0
appet             0
pe                0
ane               0
classification    0
dtype: int64
```

```python
df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    object
 1   age             400 non-null    object
 2   bp              400 non-null    object
 3   sg              400 non-null    object
 4   al              400 non-null    object
 5   su              400 non-null    object
 6   rbc             400 non-null    object
 7   pc              400 non-null    object
 8   pcc             400 non-null    object
 9   ba              400 non-null    object
 10  bgr             400 non-null    object
 11  bu              400 non-null    object
 12  sc              400 non-null    object
 13  sod             400 non-null    object
```

```
 14  pot             400 non-null    object
 15  hemo            400 non-null    object
 16  pcv             400 non-null    object
 17  wc              400 non-null    object
 18  rc              400 non-null    object
 19  htn             400 non-null    object
 20  dm              400 non-null    object
 21  cad             400 non-null    object
 22  appet           400 non-null    object
 23  pe              400 non-null    object
 24  ane             400 non-null    object
 25  classification  400 non-null    object
dtypes: object(26)
memory usage: 81.4+ KB
```

```
df.describe()
```

	id	age	bp	sg	al	su	bgr	bu	sc	
count	400.000000	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	31
mean	199.500000	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	13
std	115.614301	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	1
min	0.000000	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	
25%	99.750000	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	13
50%	199.500000	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	13
75%	299.250000	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	14
max	399.000000	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	16

Unique values in each columns

```
# Finding the unique values in the columns

for i in df_imputed.columns:
    print("************************************",i,"*********************")
    print()
    print(set(df_imputed[i].tolist()))
    print()
```

```
************************************ wc *********************

{'12300', '5700', '6400', '21600', '6900', '9200', '11300', '\t?', '4200', '12700', '12400', '12500', '10200',

************************************ rc *********************

{'4.6', '3', '5.2', '8.0', '6.4', '3.1', '2.9', '3.3', '6.2', '4.0', '6.0', '\t?', '2.1', '2.3', '3.5', '2.8',

************************************ htn *********************

{'yes', 'no'}

************************************ dm *********************

{' yes', '\tno', 'yes', '\tyes', 'no'}

************************************ cad *********************

{'yes', '\tno', 'no'}

************************************ appet *********************

{'good', 'poor'}

************************************ pe *********************

{'yes', 'no'}

************************************ ane *********************

{'yes', 'no'}

************************************ classification *********************

{'ckd\t', 'notckd', 'ckd'}
```

Abbreviations:

pcv - nwq

wc - nwq

rc - num with quot

dm- char

cad - char

classification - char

```
print(df_imputed['pcv'].mode())
print(df_imputed['wc'].mode())
print(df_imputed['rc'].mode())
```

```
0    41.0
Name: pcv, dtype: float64
0    9800.0
Name: wc, dtype: float64
0    5.2
Name: rc, dtype: float64
```

```
df_imputed['classification'].value_counts()
```

```
ckd       248
notckd    150
ckd\t       2
Name: classification, dtype: int64
```

```
df_imputed['classification'] = df_imputed['classification'].apply(lambda x:'ckd' if x=='ckd\t' else x)
```

```
df_imputed['cad'].value_counts()
```

```
no      364
yes      34
\tno      2
Name: cad, dtype: int64
```

```
df_imputed['cad'] = df_imputed['cad'].apply(lambda x:'no' if x=='\tno' else x)
```

```
df_imputed['dm'].value_counts()
```

```
no       260
yes      134
\tno       3
\tyes      2
 yes       1
Name: dm, dtype: int64
```

```
df_imputed['dm'] = df_imputed['dm'].apply(lambda x:'no' if x=='\tno' else x)
df_imputed['dm'] = df_imputed['dm'].apply(lambda x:'yes' if x=='\tyes' else x)
df_imputed['dm'] = df_imputed['dm'].apply(lambda x:'yes' if x==' yes' else x)
```

```
df_imputed['dm'].value_counts()

#PCV = Packed Cell Volume

```

```
no     263
yes    137
Name: dm, dtype: int64
```

Abbreviation

pcv - nwq

wc - nwq

rc - num with quot

dm- char

cad - char

classification - char

```
df_imputed['rc'].value_counts()
```

```
5.2    148
4.5     16
4.9     14
4.7     11
4.8     10
3.9     10
4.6      9
3.4      9
5.9      8
5.5      8
6.1      8
5.0      8
3.7      8
5.3      7
5.8      7
5.4      7
3.8      7
```

```
5.6      6
4.3      6
4.2      6
3.2      5
4.4      5
5.7      5
6.4      5
5.1      5
6.2      5
6.5      5
4.1      5
3.6      4
6.3      4
6.0      4
4.0      3
3.3      3
4        3
3.5      3
2.9      2
3.1      2
2.6      2
2.1      2
2.5      2
2.8      2
3.0      2
2.7      2
5        2
2.3      1
\t?      1
2.4      1
3        1
8.0      1
Name: rc, dtype: int64
```

```
df_imputed['rc'] = df_imputed['rc'].apply(lambda x:'5.2' if x=='\t?' else x)
```

```
df_imputed['wc'].value_counts()
```

```
9800     116
6700      10
9600       9
7200       9
9200       9
        ...
19100      1
\t?        1
12300      1
14900      1
12700      1
Name: wc, Length: 92, dtype: int64
```

'\t?' '\t6200' '\t8400'

9800

```
df_imputed['wc'] = df_imputed['wc'].apply(lambda x:'9800' if x=='\t?' else x)
df_imputed['wc'] = df_imputed['wc'].apply(lambda x:'6200' if x=='\t6200' else x)
df_imputed['wc'] = df_imputed['wc'].apply(lambda x:'8400' if x=='\t8400' else x)
```

```
df_imputed['pcv'].value_counts()
```

```
41      91
52      21
44      19
48      19
40      16
43      14
42      13
45      13
```

```
32      12
36      12
33      12
50      12
28      12
34      11
37      11
30       9
29       9
35       9
46       9
31       8
24       7
39       7
26       6
38       5
53       4
51       4
49       4
47       4
54       4
25       3
22       3
27       3
19       2
23       2
15       1
21       1
17       1
20       1
\t43     1
18       1
9        1
\t?      1
16       1
14       1
Name: pcv, dtype: int64
```

```
df_imputed['pcv'] = df_imputed['pcv'].apply(lambda x:'43' if x=='\t43' else x)
df_imputed['pcv'] = df_imputed['pcv'].apply(lambda x:'41' if x=='\t?' else x)
```

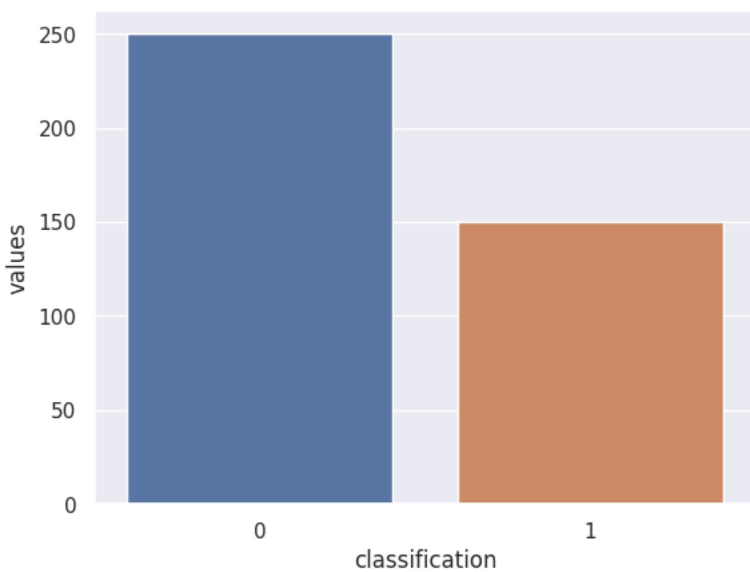
```
for i in df_imputed.columns:
    print("************************************",i,"**********************")
    print()
    print(set(df_imputed[i].tolist()))
    print()
```

```
************************************ rc *********************

{'4.6', '3', '5.2', '8.0', '6.4', '3.1', '2.9', '3.3', '6.2', '4.0', '6.0', '2.1', '2.3', '3.5', '2.8', '4.9',

************************************ htn *********************

{'yes', 'no'}

************************************ dm *********************

{'yes', 'no'}

************************************ cad *********************

{'yes', 'no'}

************************************ appet *********************

{'good', 'poor'}

************************************ pe *********************

{'yes', 'no'}

************************************ ane *********************

{'yes', 'no'}

************************************ classification *********************

{'notckd', 'ckd'}
```

Chronic Kidney Disease Vs Non Chronic Kidney Disease

```
1
2 temp = df_imputed['classification'].value_counts()
3 temp_df = pd.DataFrame({'classification': temp.index, 'values': temp.values})
4 print(sns.barplot(x = 'classification', y='values', data=temp_df))
```

```
Axes(0.125,0.11;0.775x0.77)
```




```
1 df_imputed['classification'].value_counts()
```

```
ckd       250
notckd    150
Name: classification, dtype: int64
```

```
1 df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    object
 1   age             400 non-null    object
 2   bp              400 non-null    object
 3   sg              400 non-null    object
 4   al              400 non-null    object
 5   su              400 non-null    object
 6   rbc             400 non-null    object
 7   pc              400 non-null    object
 8   pcc             400 non-null    object
 9   ba              400 non-null    object
 10  bgr             400 non-null    object
 11  bu              400 non-null    object
 12  sc              400 non-null    object
 13  sod             400 non-null    object
 14  pot             400 non-null    object
 15  hemo            400 non-null    object
 16  pcv             400 non-null    object
 17  wc              400 non-null    object
 18  rc              400 non-null    object
 19  htn             400 non-null    object
 20  dm              400 non-null    object
 21  cad             400 non-null    object
 22  appet           400 non-null    object
 23  pe              400 non-null    object
 24  ane             400 non-null    object
 25  classification  400 non-null    object
dtypes: object(26)
memory usage: 81.4+ KB
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    int64
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object
 7   pc              335 non-null    object
 8   pcc             396 non-null    object
 9   ba              396 non-null    object
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object
 17  wc              295 non-null    object
 18  rc              270 non-null    object
 19  htn             398 non-null    object
 20  dm              398 non-null    object
 21  cad             398 non-null    object
 22  appet           399 non-null    object
 23  pe              399 non-null    object
 24  ane             399 non-null    object
```

```
 25  classification  400 non-null    object
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

```
df.select_dtypes(exclude=["object"]).columns
```

```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot',
       'hemo'],
      dtype='object')
```

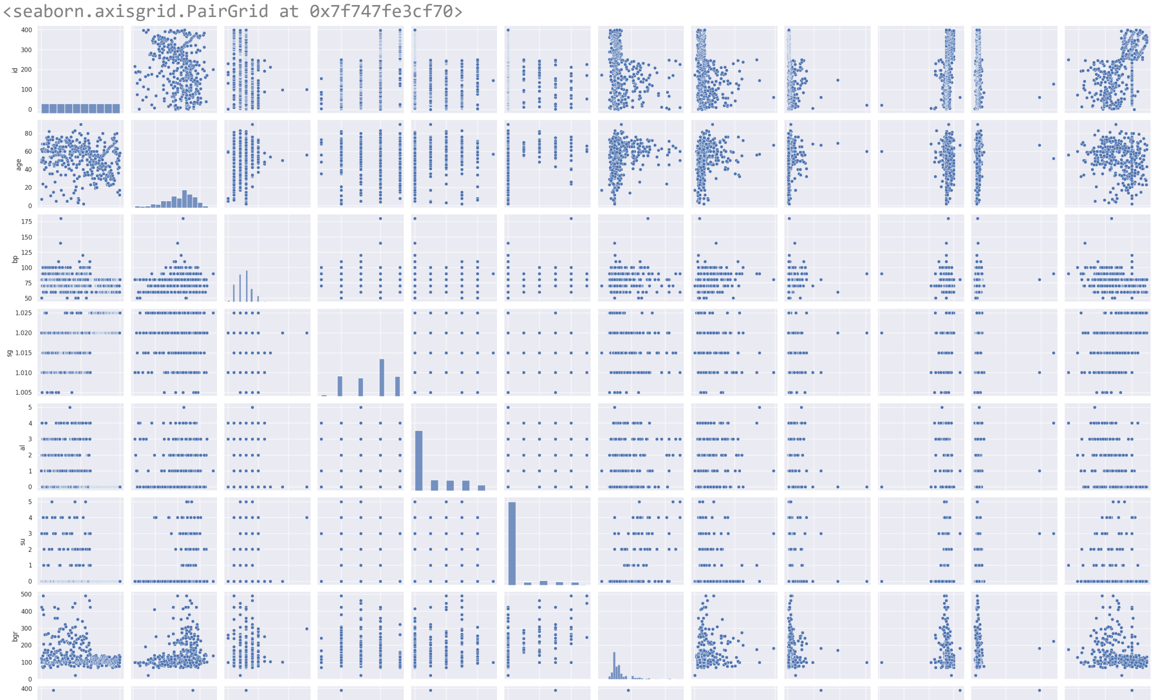
```
for i in df.select_dtypes(exclude=["object"]).columns :
    df_imputed[i] = df_imputed[i].apply(lambda x:float(x))
```

```
df_imputed.dtypes
```

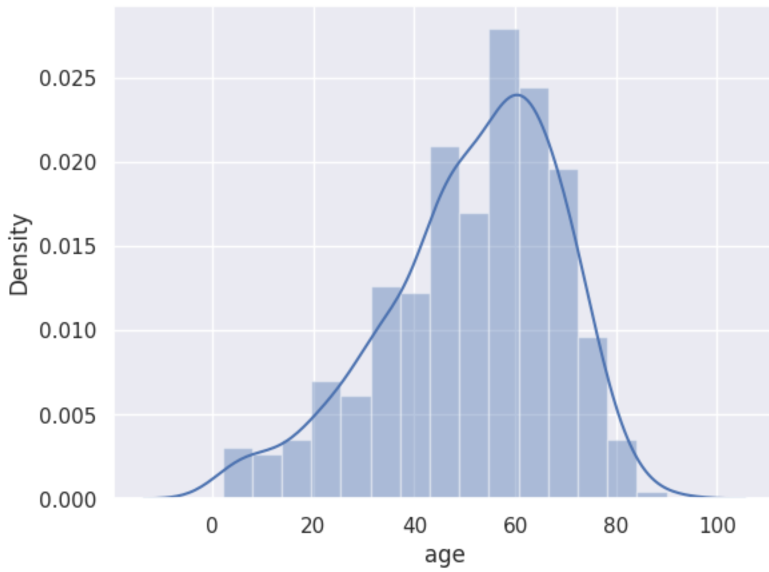
```
id                float64
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc                object
pc                 object
pcc                object
ba                 object
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv                object
wc                 object
rc                 object
htn                object
dm                 object
cad                object
appet              object
pe                 object
ane                object
classification     object
dtype: object
```

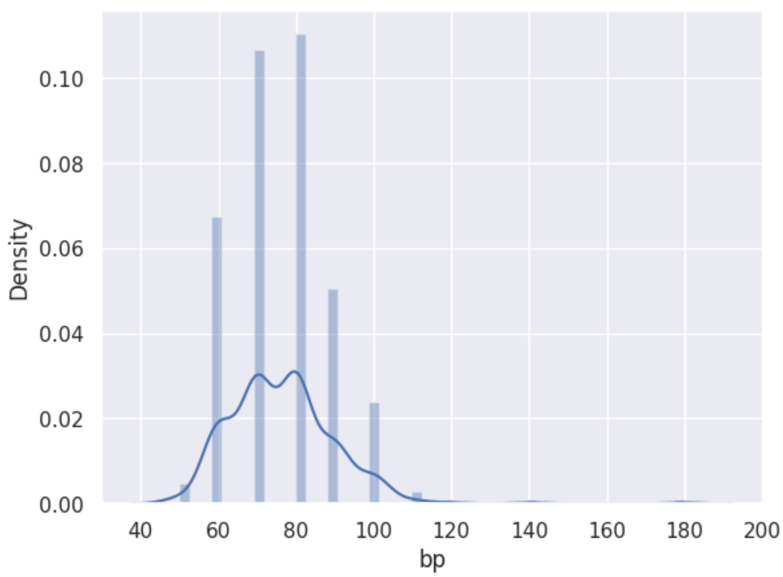
```
sns.pairplot(df_imputed)
```

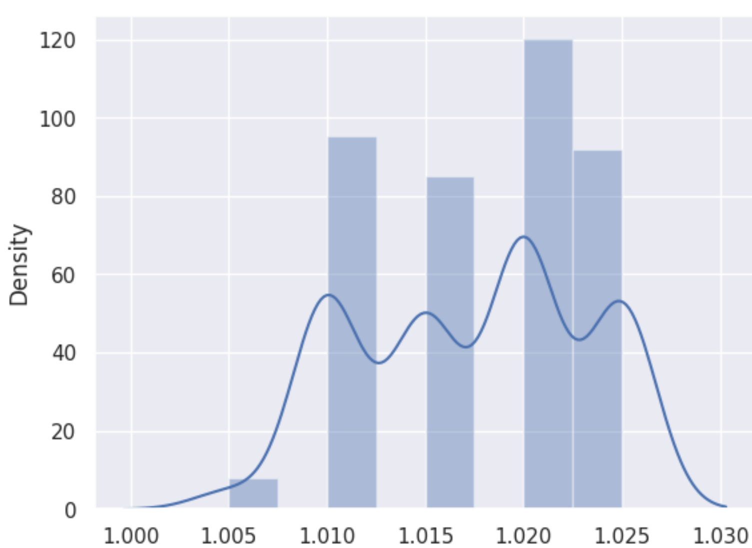
```
<seaborn.axisgrid.PairGrid at 0x7f747fe3cf70>
```



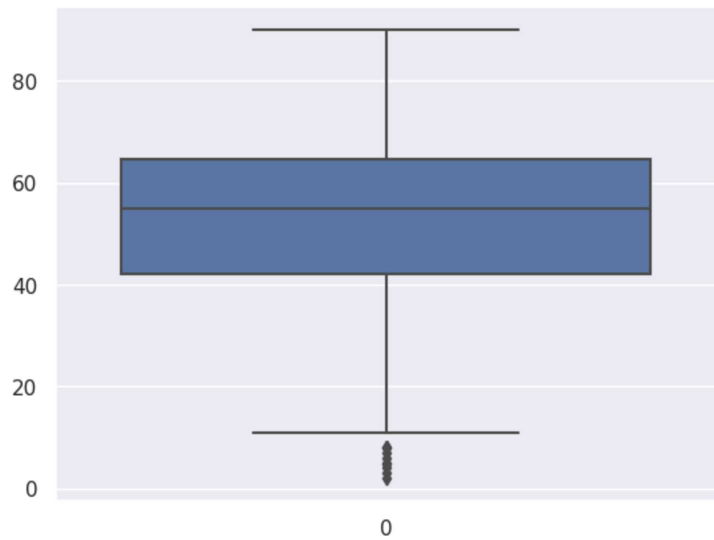
```
# Find the distribution of the data

def distplots(col):
    sns.distplot(df[col])
    plt.show()

for i in list(df_imputed.select_dtypes(exclude=['object']).columns)[1:]:
    distplots(i)
```

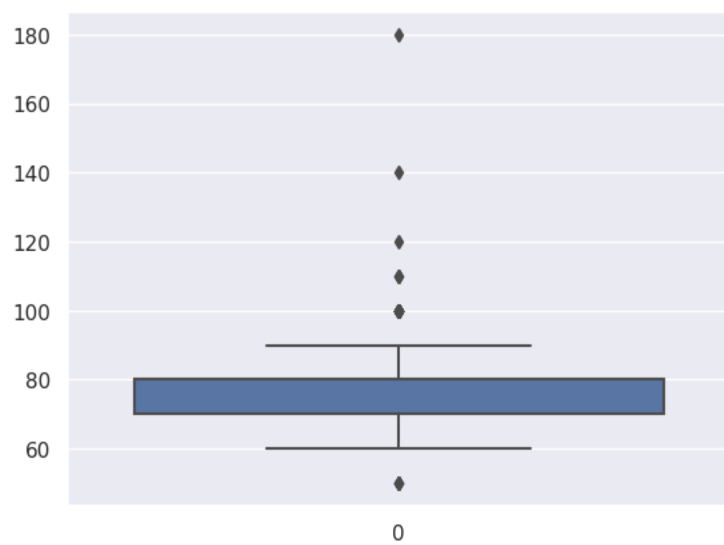

Density
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
age

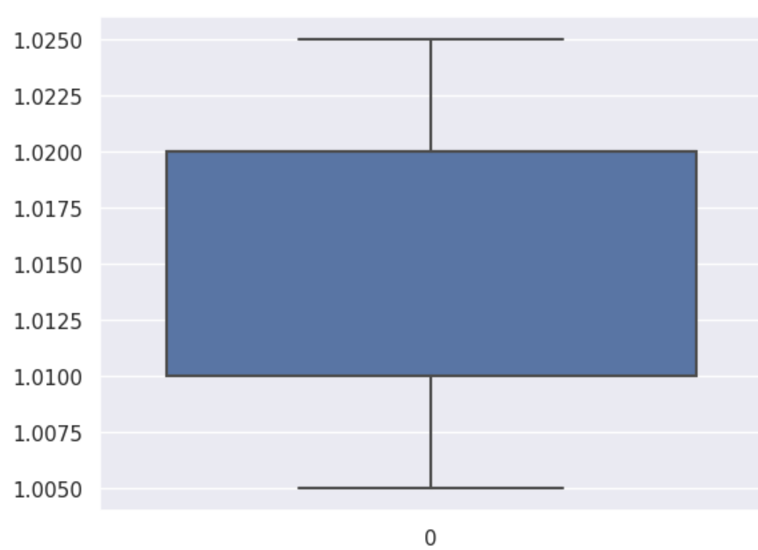

Density
0.10
0.08
0.06
0.04
0.02
0.00
40
60
80
100
120
140
160
180
200
bp

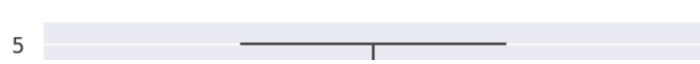

Density
120
100
80
60
40
20
0
1.000
1.005
1.010
1.015
1.020
1.025
1.030

```
# Find and remove outliers of data

def boxplots(col):
    sns.boxplot(df[col])
    plt.show()

for i in list(df_imputed.select_dtypes(exclude=['object']).columns)[1:]:
    boxplots(i)
```


80
60
40
20
0
0


180
160
140
120
100
80
60
0


1.0250
1.0225
1.0200
1.0175
1.0150
1.0125
1.0100
1.0075
1.0050
0


5

```
1 df_imputed.head(10)
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	ap
0	0.0	48.0	80.0	1.020	1.0	0.0	normal	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	g
1	1.0	7.0	50.0	1.020	4.0	0.0	normal	normal	notpresent	notpresent	...	38	6000	5.2	no	no	no	g
2	2.0	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	5.2	no	yes	no	p
3	3.0	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	p
4	4.0	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	g
5	5.0	60.0	90.0	1.015	3.0	0.0	normal	normal	notpresent	notpresent	...	39	7800	4.4	yes	yes	no	g
6	6.0	68.0	70.0	1.010	0.0	0.0	normal	normal	notpresent	notpresent	...	36	9800	5.2	no	no	no	g
7	7.0	24.0	80.0	1.015	2.0	4.0	normal	abnormal	notpresent	notpresent	...	44	6900	5	no	yes	no	g
8	8.0	52.0	100.0	1.015	3.0	0.0	normal	abnormal	present	notpresent	...	33	9600	4.0	yes	yes	no	g
9	9.0	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	...	29	12100	3.7	yes	yes	no	p

10 rows × 26 columns

```
1 df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    float64
 1   age             400 non-null    float64
 2   bp              400 non-null    float64
 3   sg              400 non-null    float64
 4   al              400 non-null    float64
 5   su              400 non-null    float64
 6   rbc             400 non-null    object
 7   pc              400 non-null    object
 8   pcc             400 non-null    object
 9   ba              400 non-null    object
 10  bgr             400 non-null    float64
 11  bu              400 non-null    float64
 12  sc              400 non-null    float64
 13  sod             400 non-null    float64
 14  pot             400 non-null    float64
 15  hemo            400 non-null    float64
 16  pcv             400 non-null    object
 17  wc              400 non-null    object
 18  rc              400 non-null    object
 19  htn             400 non-null    object
 20  dm              400 non-null    object
 21  cad             400 non-null    object
 22  appet           400 non-null    object
 23  pe              400 non-null    object
 24  ane             400 non-null    object
 25  classification  400 non-null    object
dtypes: float64(12), object(14)
memory usage: 81.4+ KB
```

Label Encoding: To convert Categorical to Numerical

```
1 # Label encodint to convert categorical values to numerical
2
```

```
from sklearn import preprocessing
"""
df_enco = df_imputed.apply(preprocessing.LabelEncoder().fit_transform)
df_enco
"""

categorical_feature = [feature for feature in df_imputed.columns if df_imputed[feature].dtype == 'O']

```

```
df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    float64
 1   age             400 non-null    float64
 2   bp              400 non-null    float64
 3   sg              400 non-null    float64
 4   al              400 non-null    float64
 5   su              400 non-null    float64
 6   rbc             400 non-null    object 
 7   pc              400 non-null    object 
 8   pcc             400 non-null    object 
 9   ba              400 non-null    object 
 10  bgr             400 non-null    float64
 11  bu              400 non-null    float64
 12  sc              400 non-null    float64
 13  sod             400 non-null    float64
 14  pot             400 non-null    float64
 15  hemo            400 non-null    float64
 16  pcv             400 non-null    object 
 17  wc              400 non-null    object 
 18  rc              400 non-null    object 
 19  htn             400 non-null    object 
 20  dm              400 non-null    object 
 21  cad             400 non-null    object 
 22  appet           400 non-null    object 
 23  pe              400 non-null    object 
 24  ane             400 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(12), object(14)
memory usage: 81.4+ KB
```

```
df_imputed['pcv'] = df_imputed['pcv'].astype('float')
df_imputed['wc'] = df_imputed['wc'].astype('float')
df_imputed['rc'] = df_imputed['rc'].astype('float')
```

```
df_imputed.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appe
0	0.0	48.0	80.0	1.020	1.0	0.0	normal	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	goo
1	1.0	7.0	50.0	1.020	4.0	0.0	normal	normal	notpresent	notpresent	...	38.0	6000.0	5.2	no	no	no	goo
2	2.0	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	5.2	no	yes	no	poo
3	3.0	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poo
4	4.0	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	goo

5 rows × 26 columns

40

```
categorical_feature = [feature for feature in df_imputed.columns if df_imputed[feature].dtype == 'O']

```

```
categorical_feature
```

```
['rbc',
 'pc',
 'pcc',
 'ba',
 'htn',
 'dm',
 'cad',
 'appet',
 'pe',
 'ane',
 'classification']
```

```
df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    float64
 1   age             400 non-null    float64
 2   bp              400 non-null    float64
 3   sg              400 non-null    float64
 4   al              400 non-null    float64
 5   su              400 non-null    float64
 6   rbc             400 non-null    object 
 7   pc              400 non-null    object 
 8   pcc             400 non-null    object 
 9   ba              400 non-null    object 
 10  bgr             400 non-null    float64
 11  bu              400 non-null    float64
 12  sc              400 non-null    float64
 13  sod             400 non-null    float64
 14  pot             400 non-null    float64
 15  hemo            400 non-null    float64
 16  pcv             400 non-null    float64
 17  wc              400 non-null    float64
 18  rc              400 non-null    float64
 19  htn             400 non-null    object 
 20  dm              400 non-null    object 
 21  cad             400 non-null    object 
 22  appet           400 non-null    object 
 23  pe              400 non-null    object 
 24  ane             400 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(15), object(11)
memory usage: 81.4+ KB
```

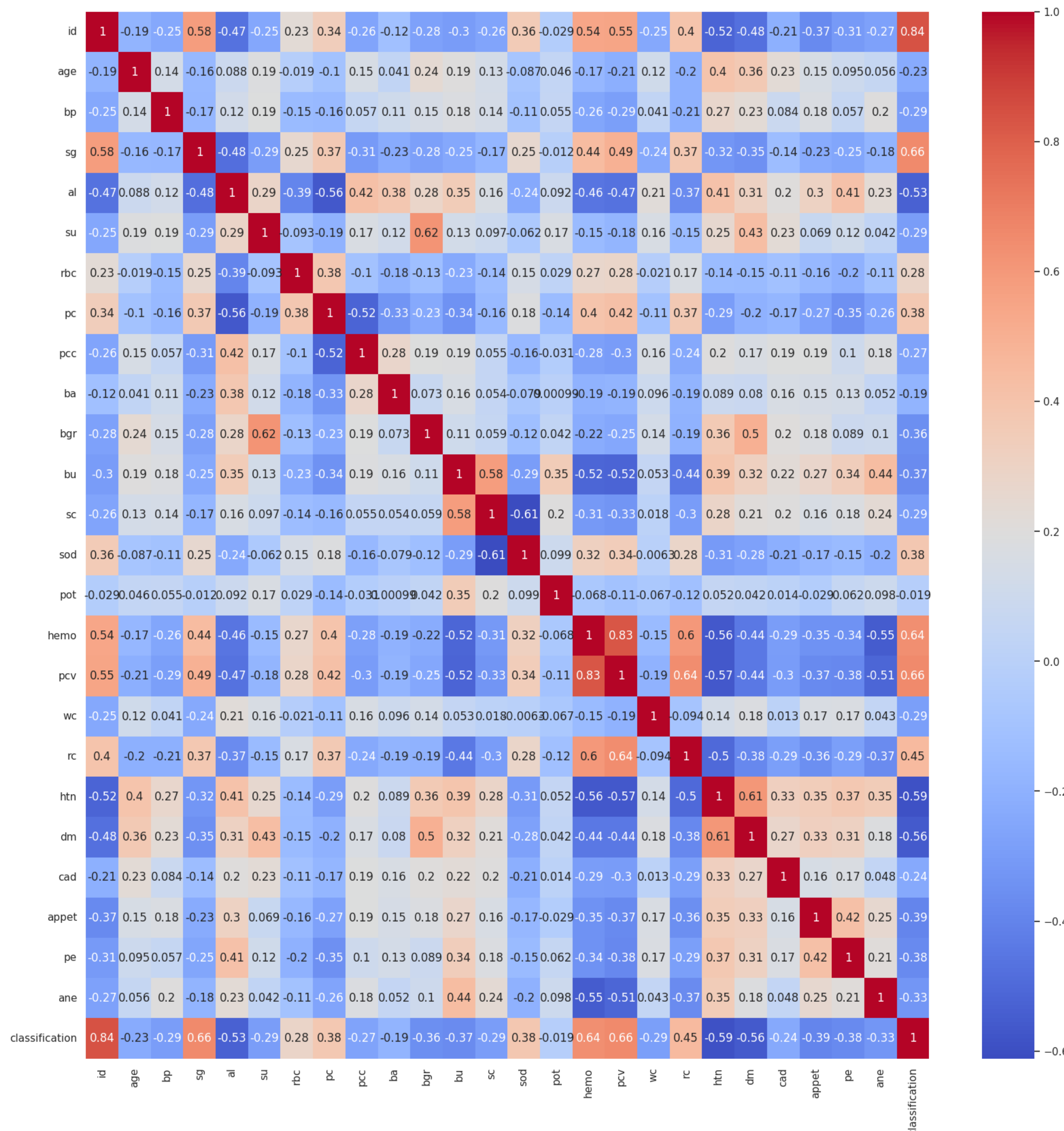
```
for i in df_imputed.select_dtypes(exclude=['float']).columns:
    df_imputed[i] = df_imputed[i].astype('category')
    df_imputed[i] = df_imputed[i].cat.codes
```

```
df_imputed.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classifica
0	0.0	48.0	80.0	1.020	1.0	0.0	1	1	0	0	...	44.0	7800.0	5.2	1	1	0	0	0	0	
1	1.0	7.0	50.0	1.020	4.0	0.0	1	1	0	0		38.0	6000.0	5.2	0	0	0	0	0	0	

```
1 # Finding the correlation
2
3 plt.figure(figsize=(20,20))
4 corr = df_imputed.corr()
5 sns.heatmap(corr, annot=True, cmap='coolwarm')
```

<Axes: >




```
1 df_imputed['classification'].value_counts()
```

```
0    250
1    150
Name: classification, dtype: int64
```

```
# split the data into independent variable and dependent variable
x = df_imputed.drop(['id','classification'], axis=1)
y = df_imputed['classification']
```

```
x.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane
0	48.0	80.0	1.020	1.0	0.0	1	1	0	0	121.0	...	15.4	44.0	7800.0	5.2	1	1	0	0	0	0
1	7.0	50.0	1.020	4.0	0.0	1	1	0	0	99.0	...	11.3	38.0	6000.0	5.2	0	0	0	0	0	0
2	62.0	80.0	1.010	2.0	3.0	1	1	0	0	423.0	...	9.6	31.0	7500.0	5.2	0	1	0	1	0	1
3	48.0	70.0	1.005	4.0	0.0	1	0	1	0	117.0	...	11.2	32.0	6700.0	3.9	1	0	0	1	1	1
4	51.0	80.0	1.010	2.0	0.0	1	1	0	0	106.0	...	11.6	35.0	7300.0	4.6	0	0	0	0	0	0

5 rows × 24 columns

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: classification, dtype: int8
```

Label Balancing with RandomOverSampler

```
# lets detect the label balance

from imblearn.over_sampling import RandomOverSampler
from collections import Counter
print(Counter(y))
```

```
Counter({0: 250, 1: 150})
```

```
ros = RandomOverSampler()
x_ros , y_ros = ros.fit_resample(x, y)
print(Counter(y_ros))
```

```
Counter({0: 250, 1: 250})
```

Feature Scaling using Standard Scaler

```
# feature scaling
# Min Max Scaler (-1,1)
# from sklearn.preprocessing import StandardScaler

scaler = MinMaxScaler((-1,1))
x = scaler.fit_transform(x_ros)
y = y_ros
```

```
x
```

```
array([[ 0.04545455, -0.53846154,  0.5       , ..., -1.        ,
         -1.        , -1.        ],
       [-0.88636364, -1.        ,  0.5       , ..., -1.        ,
         -1.        , -1.        ],
       [ 0.36363636, -0.53846154, -0.5       , ...,  1.        ,
         -1.        ,  1.        ],
       ...,
       [-0.54545455, -0.84615385,  1.        , ..., -1.        ,
         -1.        , -1.        ],
       [-0.40909091, -0.84615385,  1.        , ..., -1.        ,
         -1.        , -1.        ],
       [ 0.40909091, -0.84615385,  0.5       , ..., -1.        ,
         -1.        , -1.        ]])
```

```
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
495    1
496    1
497    1
498    1
499    1
Name: classification, Length: 500, dtype: int8
```

This is all about the pre-processing approach

```
x.shape
```

```
(500, 24)
```

```
# Applying PCA method to reduce non-significant variables and values (if any)

from sklearn.decomposition import PCA

# it means the machine(sklearn) choose the minimum number of PCA such that 95% of the variance is retained

pca = PCA(0.95)
x_pca = pca.fit_transform(x)
print(x.shape)
print(x_pca.shape)
```

```
(500, 24)
(500, 15)
```

```
# split the data into training and test for building the model and prediction with test dataset
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_pca, y, test_size=0.2, random_state=5)
```

Model Building

MLP - multilayer perceptron (Neural Network)

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import Model
from keras.optimizers import Adam
```

creating the model

```
def model():
    classifier = Sequential()
    classifier.add(Dense(15, input_shape=(x_train.shape[1],),activation='relu'))
    classifier.add(Dropout(0.2))
    classifier.add(Dense(15, activation='relu'))
    classifier.add(Dropout(0.4))
    classifier.add(Dense(1, activation='sigmoid'))
    classifier.compile(optimizer ='adam', loss='binary_crossentropy',metrics =['accuracy'])

    return classifier

```

```
model = model()
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 15)                240

 dropout_4 (Dropout)         (None, 15)                0

 dense_7 (Dense)             (None, 15)                240

 dropout_5 (Dropout)         (None, 15)                0

 dense_8 (Dense)             (None, 1)                 16

=================================================================
Total params: 496
Trainable params: 496
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=50, verbose=1)
```

```
13/13 [==============================] - 0s 6ms/step - loss: 0.1000 - accuracy: 0.9850 - val_loss: 0.0318 - va
Epoch 33/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0920 - accuracy: 0.9725 - val_loss: 0.0295 - va
Epoch 34/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0775 - accuracy: 0.9800 - val_loss: 0.0275 - va
Epoch 35/50
13/13 [==============================] - 0s 7ms/step - loss: 0.0754 - accuracy: 0.9850 - val_loss: 0.0255 - va
Epoch 36/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0837 - accuracy: 0.9725 - val_loss: 0.0239 - va
Epoch 37/50
13/13 [==============================] - 0s 8ms/step - loss: 0.0822 - accuracy: 0.9725 - val_loss: 0.0223 - va
Epoch 38/50
13/13 [==============================] - 0s 5ms/step - loss: 0.0770 - accuracy: 0.9850 - val_loss: 0.0206 - va
Epoch 39/50
13/13 [==============================] - 0s 7ms/step - loss: 0.0618 - accuracy: 0.9850 - val_loss: 0.0192 - va
Epoch 40/50
13/13 [==============================] - 0s 5ms/step - loss: 0.0837 - accuracy: 0.9700 - val_loss: 0.0180 - va
Epoch 41/50
13/13 [==============================] - 0s 5ms/step - loss: 0.0536 - accuracy: 0.9875 - val_loss: 0.0169 - va
Epoch 42/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0575 - accuracy: 0.9875 - val_loss: 0.0157 - va
Epoch 43/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0582 - accuracy: 0.9875 - val_loss: 0.0147 - va
Epoch 44/50
13/13 [==============================] - 0s 7ms/step - loss: 0.0448 - accuracy: 0.9900 - val_loss: 0.0137 - va
Epoch 45/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0576 - accuracy: 0.9800 - val_loss: 0.0129 - va
Epoch 46/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0611 - accuracy: 0.9900 - val_loss: 0.0120 - va
Epoch 47/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0588 - accuracy: 0.9850 - val_loss: 0.0111 - va
Epoch 48/50
13/13 [==============================] - 0s 5ms/step - loss: 0.0567 - accuracy: 0.9850 - val_loss: 0.0103 - va
Epoch 49/50
13/13 [==============================] - 0s 6ms/step - loss: 0.0423 - accuracy: 0.9950 - val_loss: 0.0096 - va
Epoch 50/50
```

```
y_pred = model.predict(x_test)
y_pred = (y_pred>0.5)
y_pred
```

```
4/4 [==============================] - 0s 3ms/step
array([[False],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
```

```
        [False],
        [ True],
        [False],
        [ True],
        [False],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [False],
        [False],
        [ True],
        [False],
        [False],
        [ True],
        [ True],
        [False],
        [False],
        [ True],
        [ True],
        [False],
        [False],
        [ True],
        [False],
        [False],
```

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
print(confusion_matrix(y_test, y_pred))
```

```
[[50  0]
 [ 0 50]]
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100
```

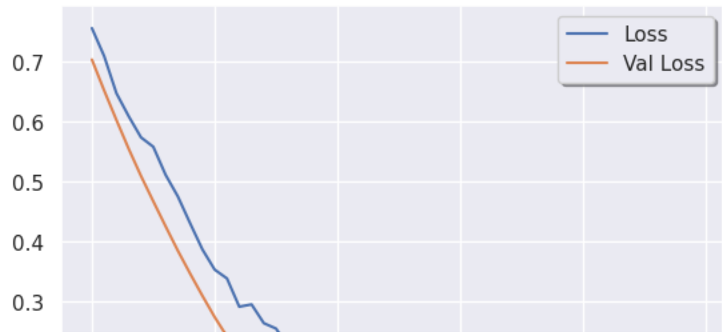
Accuracy is 1 (100%)

```
print(accuracy_score(y_test, y_pred))
```

```
1.0
```

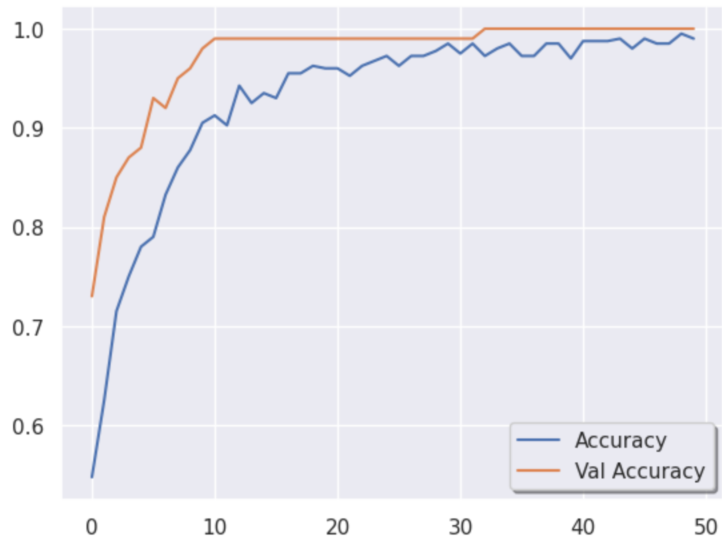
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(('Loss', 'Val Loss'),
           loc='upper right', shadow=True)
```

```
<matplotlib.legend.Legend at 0x7f7478a99a60>
```




```
plt.plot(history.history['accuracy'], label='test')

plt.plot(history.history['val_accuracy'])
plt.legend(('Accuracy', 'Val Accuracy'),
           loc='lower right', shadow=True)
```

```
<matplotlib.legend.Legend at 0x7f7481939340>
```



0s completed at 3:39 PM